Low Power SRAM Design for PICo

Andrew Ashworth,

Jonathan Chen,

Matt Williams

ECE 432 – Fall 2008  
University of Virginia  
[jc4ec@virginia.edu](mailto:jc4ec@virginia.edu) [mcw8d@virginia.edu](mailto:mcw8d@virginia.edu) [ala4a@virginia.edu](mailto:ala4a@virginia.edu)

**ABSTRACT**

This report for PICo documents the progress and design of a low power 1 Mb embedded SRAM. Simulations and results are compiled from Cadence using Spectre. By using a drowsy cache, and divided bit-lines, low power consumption can be achieved.

# INTRODUCTION

The design examined in this report is the embedded SRAM built using AMI 0.6 um technology. Using Ocean, Cadence, and several references, a suitable SRAM architecture and layout was designed to satisfy design restrictions and to meet certain design metrics. In order to prove that the SRAM designed meets and exceeds expectations, simulation results in all process corners, layout diagrams, and schematics will be presented to PICo.

# METRICS AND REQUIREMENTS

In order to meet the requirements set by PICo, the designed SRAM must meet certain design metrics as measurement of success. Of the two design applications suggested, the SRAM outlined focuses on low power as its primary concern. To measure success, the following equation is applied:

(total power)2 \* delay \* area

By minimizing the value obtained from the equation, success of the SRAM can be quantified. The units of measure for each of the metrics are as follows:

total power = mWatts2

delay = nanoseconds

area = mm2

In addition to satisfying the design metrics, a lower power SRAM, as specified by PICo, must be 1 Mb large, have a word size of thirty-two bits, an output of thirty-two bits, and maintain one read or one write access per cycle.

# ARCHITECTURE

## Clock

The design of the controlling clock signals has gone through three iterations. The original control consisted of an inverter chain, used to produce a delayed, inverted clock. The delayed, inverted clock and the original clock were then used to control pre-charge and decode enable. The next iteration consisted of a T flip-flop that produced a second clock signal that was half the frequency of the input clock. While this control structure provided four separate stages for SRAM activities, there is overlap between signals that caused numerous timing difficulties. The final clock

control is a two-phase non-overlapping clock generator. This produces a clock and clock bar that will never have overlapping

logic high. One benefit of this structure is that it provides three separate clocks, the original, a slightly delayed original and a delayed, inverted clock.

CLK

CLK1

CLK2

4:16 bit

Enable bit

Enable bit

4:16 bit

**Figure 1 : Two-Phase Non-Overlapping Clock Generator**

## Array Architecture

Designing the array architecture to be the most efficient for power is very important due to the relatively large size of the proposed SRAM. The capacitances associated with the main data and control lines, i.e. the bit-lines and word-lines, have to be considered. To do this, the final design requires that the 1Mb SRAM array be broken up into smaller blocks, each holding a fixed number of bit-cells. There are several advantages to using this block architecture. First of all, designing such a large SRAM would be schematically difficult if only a single block is used. Also, splitting up the SRAM would allow the use of global word-lines and bit-lines, which can be used to reduce capacitances. Lastly, by using 2n blocks in the array, binary addressing is simplified.

After making the design decision to use 25 or thirty-two blocks to limit the capacitances present in a single block, it was determined that a single block would be enabled by an enable bit coming from a global 5:32 decoder driven by the five most significant bits of the fifteen bit address (see Figure 2).

5:32

<4:7>

<0:3>

<8:11>

<12:15>

<16:19>

<20:23>

<24:27>

<28:31>

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

A14A13A12A11A10

Transmission Gates

Word-line Enable Bit

**Figure 2 : Array Architecture with Global Decoder**

The same global world-line enable bit will also be used to drive the enable bits of transmission gates placed at the data output of each block. Therefore, the output is only chosen from one of the thirty-two blocks.

As this SRAM array relies heavily on large decoder logic, to try and minimize the delay and power consumption, an efficient two-stage decoder design is implemented

## Block Architecture

Since the SRAM array uses thirty-two blocks for 1 Mb of storage, each block has 32768 bit-cells, meaning the block is 256 bit-cells tall and 128 bit-cells wide.

### The Bit-Cell

The bit-cell used by the SRAM architecture is the standard six transistor bit-cell sized for robust reads and writes. The sizes for the access transistors and cross-coupled inverters are obtained through a series of calculations (which can be found attached to this report) to find the ratio between transistors vulnerable to a fight during a read and a write (see Figure 4)**.**

Local BL

Local BLB

Local WL

Local WL

Transistors use 0.5 um technology. Sizes shown represent widths of devices.

1.5um

1.5um

1.5um

4.5um

4.5um

1.5um

4.5um

1.5um

1.5um

1.5um

4.5um

**Figure 4 : The 6T Bit-Cell**

### Word-lines and Bit-lines

The block itself will have a unique global and local bit-line and word-line system. The division sizes for the local bit-lines were determined by the calculations which can be found attached to this report [1]. This system helps manage the large capacitances that would otherwise be present in large continuous bit-lines and word-lines.

The local word-lines are driven by the output from a 4:16 decoder, which is enabled by the output of another 4:16 decoder, which is enabled by the output from the 5:32 decoder described in the array architecture. To minimize the delays in the decoders, sized inverters are added in between the decoders (see Figure 5).

To local word-line

Numbers shown by inverters are ratios relative to minimum sized inverter

1

4

A9A8A7A6

A5A4A3A2

Enable bit from 5:32 Decoder

4:16

4:16

1

4

1

4

Enable bit

Enable bit

4:16 bit

4:16 bit

**Figure 5 : Block Decoder Logic**

The local bit-lines are fed from the pass transistors in which the word-lines enable signal is driven into. The bit-line signals are organized in a hierarchical manner so that at one time, only sixteen local pass transistors on the entire global bit-line, will be on (see Figure 6).

from Figure 5

Global Bit-Line

Global Bit-Line Bar

**Figure 6 : Divided Word-line and Bit-lines with Bit-Cells**

To select a sixteen bit word in a block, a 2:4 decoder with A1A0 address input will output an enable bit, going to transmission gates on the bit-lines and bit-line bars. Another transmission gate will also be on the bit-line and bit-line bar, with its enable bit being the block enable bit from the 5:32 decoder. Each of the four outputs from the decoder will feed sixteen columns. Therefore, for any operation, only sixteen columns (one word) will be selected at one time (see Figure 7).

To Sense-Amplifiers

128 Bit-Cells (4 Words)

Word 1

Word 2

Word 3

Word 4

A1A2

Bit-Lines/Bit-Line Bars from Word 1

Enable Bits to Word 2, 3, 4

Enable bit from 5:32 decoder

To Sense-Amplifiers

T-Gate

T-Gate

2:4

**Figure 7 : Block Output Logic**

### Sense-Amplifier and Drivers

The differential sense-amplifier is another key component to the functionality of the block. The sense-amplifier takes the signal from bit-line and bit-line bar when read enable is asserted, and then the sense-amplifier determines which of the signals is a zero by its decay, and outputs the value of the bit-line based on that function (see Figure 7).

SA-Enable

SA-Enable

All devices use minimum sizes.

Bit-Line

SA-Enable

Bit-Line Bar

Out

**Figure 8 : Differential Sense Amp Schematic**

In order to make the bit-cell block functional, control signals needed to be provided. To do this, peripheral logic was implemented so that reads and writes could be performed while saving as much power as possible to meet the design metrics. As a result, through the use of minimum sized transistors, the write, word-line access, pre-charge, and differential sense-amplifier control drivers are designed using pass gates, and simple combinational logic utilizing the provided CLK (see Figure 8).

All devices are minimum sized unless noted.

CLK1

Write

Write Enable

DataIn

4x

4x

4x

Write Enable

BLB

BL

Precharge Logic

Decoder Enable

Write Driver

Write Enable

CLK1

Word-line Enable

PRE

PRE

PRE

VDD

BL

BLB

CLKB

Write

CLKB

Read

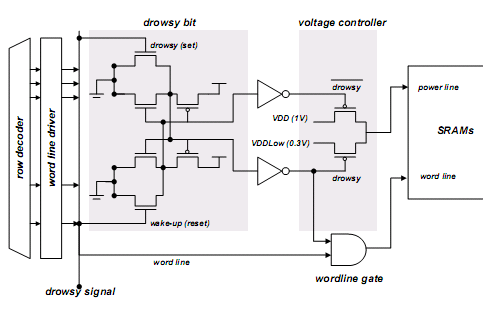
Decoder Enable

(to 5:32 decoder)

**Figure 8 : Control Bit Drivers**

# DROWSY CACHE

Adjusting VDD is one of the most effective ways to reduce power consumption in a circuit. The drowsy cache implementation is simply a way to dynamically reduce VDD based on whether or not we are accessing a block in our SRAM. If the block is being accessed, supply voltage would be turned to 2.5V. If it is low, the voltage would be lowered to 750mV, the minimum necessary to maintain reasonable noise margins and hold the value stored inside the 6T cell. The drowsy bit itself is simply stored by cross-coupled inverters, and is written by activating the active-high wake-up signal. This bit activates one of two power transistors that pass 2.5V or 750mV respectively. This adds a little bit of delay, but reduces leakage power by many orders of magnitude.



**Figure 9 : Drowsy Cache Schematic [3]**

# CHALLENGES

## Ocean vs. Cadence

In the initial designs of the SRAM, a proof of concept schematic was created in net-lists and simulated in Ocean using Spectre. This early design review was a simulation of a single bit-cell, with a small amount of peripheral logic.

However, because of the inability of Ocean to do layout simulations, all of the net-lists needed to be transferred into Cadence schematic files so LVS checking could be completed. This process alone consumed a lot of time that could have otherwise been spent on further improving the design of the SRAM. Nonetheless, an improved design was completed on time for the second design review.

## Global and Local Lines

In addition to time constraints due to converting net-lists into schematics, attention had to be brought to the fact that capacitances on the bit-lines, bit-line bars, and word-lines would be enormous without proper designing. To do this, a solution of using global and local lines was applied. Doing this allows the selection of only a few bit-cells in the block, further reducing the power required to access a block.

As outlined in Section 3.2.2: Word-lines and Bit-lines, a decision needed to be made to determine the size and design of the global/local word-line and bit-line system. By using the calculations as noted in the section, a solution with support for the decision was found.

In the entire context of the SRAM array, only the bit-lines and word-lines associated with thirty-two bits (reading/writing a single word) would be put through the pre-charge and access cycles. As opposed to charging all bit-lines in the entire SRAM, this is good power saving design.

## Layout Concerns

Because of the relatively large area that a 1 Mb SRAM would occupy, designing and completing the layout for the whole SRAM was a challenge of its own due to time constraints. After the individual components of the SRAM are laid out, the components have to be pieced together in a logical fashion, so that the SRAM takes up the least amount of space from hardware prospective. The results and pictures of the layouts are detailed in Section 6: Simulations and Results.

## Drivers and Control Signals

The inputs to the SRAM are extremely sensitive to the timing of the reads and writes. Therefore, we had to make sure that our control signals were accurate enough to perform a read and write. Rising and fall times of the non-ideal signals proved challenging to work with. To work around these problems, a two-phase non-overlapping clock generator was used to provide a reliable clock for the system. Furthermore, buffers were added at the outputs of various drivers to speed up rising and fall times.

However, one control signal, the sense amplifier enable bit, proved too complicated to implement given the time constraints, and a buffered ideal input was used instead. The sense amplifier control bit needed to assert in a short pulse, something that proved very difficult to create with the use of non-ideal signals produced by drivers.

# SIMULATIONS AND RESULTS

Due to the impracticality of simulating the entire 1 Mb SRAM array, a smaller scale simulation of the write and read of one bit-cell was performed to prove functionality. The one bit-cell is simulated in a array model with extracted capacitances (calculation can be found attached to report).

Simulations of the SRAM performing a write zero and read zero were conducted at zero, twenty-seven, fifty degrees Centigrade. Within each of these three temperatures, simulations were also done at the five process corners, SS, SF, TT, FS, and FF.

To tweak the timing of the SRAM, simulations were first performed starting with fifty degrees at SS (worst-case), and adjusted the timing until just before the SRAM would fail to function properly. From there, all other simulations were run using the tweaked timing.

As a result of the simulations conducted on the SRAM, the following metrics were obtained. Calculations for each of the metrics are attached along with the simulation result graphs.

Total Power: 9.291 mW

Delay: 35.327 ns

Area: 478.19 mm2 (bit-cell array)

(total power)2 \* delay \* area = **1458250.615**

# REFERENCES

1. Bowman, B., Debray, S. K., and Peterson, L. L. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst., 15,* 5 (Nov. 1993), 795-825.
2. Karandikar, A., Parhi, K. K. (1998), Lower power SRAM design using hierarchical divided bit-line approach. Proceedings of the 1998 IEEE International Conference on Computer Design, 82-88.
3. Flautner, K. and Kim, N.S. and Martin, S. and Blaauw, D. and Mudge, T. Drowsy Caches: simple techniques for reducing leakage power. Proceedings of the 2002 Annual International Symposium on Computer Architecture (2002), 148-157